



⑮ BUNDESREPUBLIK  
DEUTSCHLAND



DEUTSCHES  
PATENT- UND  
MARKENAMT

⑫ Off nl gungsschrift  
⑩ DE 199 34 515 A 1

⑤ Int. Cl.<sup>7</sup>:  
G 06 F 12/08

② Aktenzeichen: 199 34 515.5  
③ Anmeldetag: 22. 7. 1999  
④ Offenlegungstag: 27. 1. 2000

DE 199 34 515 A 1

③ Unionspriorität:  
09/122349 24. 07. 1998 US  
⑦ Anmelder:  
Intel Corp., Santa Clara, Calif., US  
⑦ Vertreter:  
Zenz, Helber, Hosbach & Partner, 45128 Essen

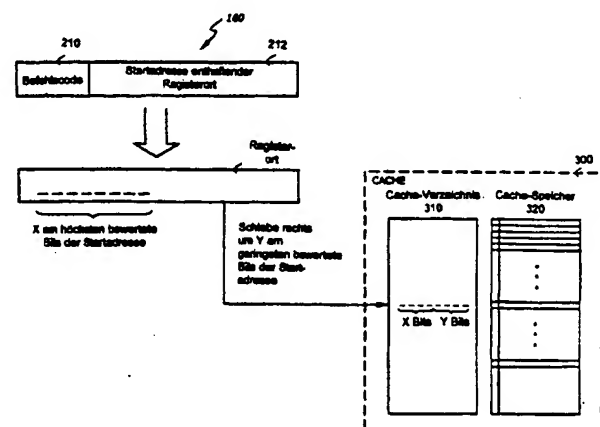
⑦ Erfinder:  
Hacking, Lance, Portland, Oreg., US; Thakkar,  
Shreekant, Portland, Oreg., US; Huff, Thomas,  
Portland, Oreg., US; Pentkovski, Vladimir, Folsom,  
Calif., US; Hsieh, Hsien-Cheng E., Gold River, Calif.,  
US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

⑤ Verfahren und Einrichtung zum Durchführen von Cache-Segment-Flush- und Cache-Segment-Invalidier-Operationen

⑤ Das Verfahren bzw. die Einrichtung zum Durchführen von Cache-Segment-Flush- und Cache-Segment-Invalidier-Operationen verwendet spezielle Befehle, wobei durch Ausführen eines einzelnen dieser Befehle durch einen Prozessor Daten in einem Segment eines Cache-Speichers invalidisiert bzw. einer Flush-Operation unterzogen werden. Der einzelne Befehl (160) weist einen Befehlscode (210) und einen ein Register spezifizierenden Abschnitt (212) auf. Das von dem Befehl spezifizierte Register enthält Daten, aus denen eine Startadresse eines Segments des Cache-Speichers (300) bestimmt wird. Die in dem Cache-Segment enthaltenen Daten werden anschließend invalidiert oder einer Flush-Operation unterzogen, d. h. aus dem Cache-Speicher in einen außerhalb des Cache-Speichers liegenden Speicherbereich kopiert bzw. geschrieben.



DE 199 34 515 A 1

Die Erfindung bezieht sich auf Verfahren und Einrichtungen eines Computersystems, welche das Invalidieren und/oder die Flush-Operation eines Abschnitts eines Cache-Speichers erleichtern.

Die Verwendung eines Cache-Speichers in einem Computersystem ermöglicht die Verringerung der Speicherzugriffszeit. Die grundsätzliche Idee der Cache-Organisation besteht darin, daß durch Halten der am häufigsten zugegriffenen Befehle und Daten in dem schnellen Cache-Speicher die durchschnittliche Speicherzugriffszeit sich der Zugriffszeit des Cache-Speichers annähert. Um einen optimalen Kompromiß zwischen der Cache-Größe und Leistung zu erreichen, implementieren typische Computersysteme eine Cache-Hierarchie, d. h. verschiedene Ebenen von Cache-Speichern. Die verschiedenen Ebenen von Cache-Speichern korrespondieren mit unterschiedlichen Abständen von dem Kern des Prozessors. Je näher der Cache dem Prozessor ist, desto schneller ist der Datenzugriff. Je näher der Cache dem Prozessor ist, desto kostenaufwendiger ist er jedoch zu implementieren. Im Ergebnis ist der Cache umso schneller und kleiner, je näher die Cache-Ebene ist.

Eine Cache-Einheit ist üblicherweise zwischen dem Prozessor und dem Hauptspeicher angeordnet; sie umfaßt üblicherweise eine Cache-Steuereinrichtung (Cache-Controller) und einen Cache-Speicher, wie beispielsweise einen statischen Speicher mit wahlfreiem Zugriff (SRAM). Die Cache-Einheit kann auf dem gleichen Chip wie der Prozessor enthalten oder als separate Komponente vorhanden sein. Alternativ kann die Cache-Steuereinrichtung auf dem Chip des Prozessors enthalten und der Cache-Speicher durch externe SRAM-Chips ausgebildet sein.

Die Leistung des Cache-Speichers wird häufig anhand seines Trefferverhältnisses gemessen. Wenn der Prozessor auf den Speicher Bezug nimmt und Daten in seinem Cache vorfindet, wird dies als Treffer bezeichnet. Wenn die Daten nicht in dem Cache gefunden werden, so sind sie in dem Hauptspeicher, was als Fehlversuch gezählt wird. Wenn ein Fehlversuch auftritt, dann wird eine Zuweisung eines Eintrags getroffen, der von der Adresse des Zugriffs indiziert wird. Bei dem Zugriff kann es sich um ein baden von Daten in den Prozessor oder ein Speichern von Daten aus dem Prozessor in den Speicher handeln. Die cache-gespeicherten Informationen werden von dem Cache-Speicher gehalten, bis sie nicht mehr benötigt werden, ungültig werden oder durch andere Daten ersetzt werden, in welchen Fällen die Zuweisung des Cache-Eintrags aufgehoben wird.

Wenn andere Prozessoren oder Systemkomponenten Zugriff auf den Hauptspeicher haben, wie das beispielsweise bei einem DMA-Controller der Fall ist, und der Hauptspeicher überschrieben werden kann, muß die Cache-Steuereinrichtung den betroffenen Cache darüber informieren, daß die Daten in dem Cache ungültig sind, sofern sich die Daten in dem Hauptspeicher geändert haben. Eine solche Operation ist als Cache-Invalidieren bekannt. Wenn die Cache-Steuereinrichtung eine Rückschreib-Strategie implementiert und bei einem Cache-Treffer die Daten nur aus dem Prozessor in den Cache schreibt, müssen die Cache-Inhalte unter speziellen Bedingungen zu dem Hauptspeicher übertragen werden. Dies ist beispielsweise der Fall, wenn das DMA-Chip Daten aus dem Hauptspeicher zu einer Peripherieeinheit überträgt, aber die aktuellen Werte nur in einem SRAM-Cache gespeichert sind. Diese Art der Operation ist als Cache-Flush-Operation (Cache-Spülen) bekannt.

Gegenwärtig werden derartige Invalidier- und/oder Flush-Operationen für eine zugehörige Cache-Zeile automatisch von Hardware ausgeführt. Für bestimmte Situatio-

nen wurde Software entwickelt, um den Cache-Speicher zu invalidieren und/oder zu spülen (flush). Gegenwärtig umfassen derartige Software-Techniken die Verwendung eines Befehls, welcher an dem gesamten Cache-Speicher, der zu dem Prozessor gehört, von welchem der Befehl herrührt, operiert. Jedoch erfordern derartige Invalidier- und/oder Flush-Operationen eine große Zeitdauer für ihren Abschluß und bieten keine Granularität oder Steuermöglichkeit für den Benutzer, um spezielle Daten oder Abschnitte von Daten aus dem Cache zu invalidieren und/oder zu spülen, während andere Daten innerhalb des Cache-Speichers intakt bleiben. Wenn eine Flush-Operation nur an dem gesamten Cache-Speicher operieren kann, führt das zu einer Inflexibilität und mindert die Systemleistung. Außerdem kann eine Datenverfälschung auftreten, wenn eine Cache-Invalidierungs-Operation nur an dem gesamten Cache möglich ist. Aufgabe der Erfindung ist es, die genannten Nachteile zu vermeiden.

Diese Aufgabe wird erfindungsgemäß durch ein Computersystem mit den Merkmalen des Patentanspruchs 1 bzw. 7, einen Prozessor mit den Merkmalen des Patentanspruchs 13 bzw. 18, ein Verfahren mit den Merkmalen des Patentanspruchs 24 bzw. 29 bzw. eine computer-lesbare Einrichtung mit den Merkmalen des Patentanspruchs 35 bzw. 36 gelöst.

Die Erfindung umfaßt ein Verfahren und eine Einrichtung, die Befehle zum Durchführen von Cache-Speicher-Invalidier- und Cache-Speicher-Flush-Operationen in ein Computersystem einbringt. Bei einem Ausführungsbeispiel umfaßt das Computersystem einen Cache-Speicher mit einer Vielzahl von Cache-Zeilen, die jeweils Daten speichern, und einen Speicherbereich zum Speichern eines Datenoperanden. Eine Ausführungseinheit ist mit dem Speicherbereich gekoppelt und operiert in Abhängigkeit vom Empfang eines einzelnen Befehls an Datenelementen in dem Datenoperanden, um Daten in einem vorgegebenen Abschnitt der Mehrzahl von Cache-Zeilen zu invalidieren.

Vorteilhafte Weiterbildungen sind in den Unteransprüchen gekennzeichnet.

Im folgenden wird die Erfindung anhand von in der Zeichnung dargestellten Ausführungsbeispielen näher beschrieben.

Die Zeichnung umfaßt folgende Figuren:

Fig. 1 veranschaulicht ein Beispielcomputersystem.

Fig. 2 veranschaulicht ein Ausführungsbeispiel des Formats einer Cache-Steuereingabe 160, der bei einem Ausführungsbeispiel der Erfindung vorgesehen ist.

Fig. 3 veranschaulicht die grundsätzliche Betriebsweise der Cache-Steuertechnik gemäß einem Ausführungsbeispiel der Erfindung.

Fig. 4A veranschaulicht ein Ausführungsbeispiel der Betriebsweise des Cache-Segment-Invalidier-Befehls 162.

Fig. 4B veranschaulicht ein Ausführungsbeispiel der Betriebsweise des Cache-Segment-Flush-Befehls 164.

Fig. 4C veranschaulicht ein Ausführungsbeispiel eines Cache-Segment-Flush- und Invalidier-Befehls 166.

Fig. 5A ist ein Ablaufdiagramm, das ein Ausführungsbeispiel des Cache-Segment-Invalidier-Prozesses gemäß der vorliegenden Erfindung veranschaulicht.

Fig. 5B ist ein Ablaufdiagramm, das ein Ausführungsbeispiel des Cache-Segment-Flush-Prozesses gemäß der vorliegenden Erfindung veranschaulicht.

In der folgenden Beschreibung werden zahlreiche spezielle Details angegeben, um ein besseres Verständnis der Erfindung zu erreichen. Es ist jedoch klar, daß die Erfindung auch ohne diesen speziellen Details ausgeführt werden kann. An anderen Stellen werden gut bekannte Schaltungen, Strukturen und Techniken nicht im Detail gezeigt, um die Erfindung nicht zu verdunkeln.

Fig. 1 veranschaulicht ein Computersystem, welches die Prinzipien der Erfindung implementieren kann. Das Computersystem 100 umfaßt einen Prozessor 105, eine Speichereinrichtung 110 und einen Bus 115. Der Prozessor 105 ist mit der Speichereinrichtung 110 über den Bus 115 gekoppelt. Die Speichereinrichtung 110 steht stellvertretend für einen oder mehrere Mechanismen zum Speichern von Daten. Beispielsweise kann die Speichereinrichtung 110 einen Nur-Lese-Speicher (ROM), einen Speicher mit wahlfreiem Zugriff (RAM), ein Magnetplattenspeichermedium, ein optisches Speichermedium, Cache-Speicherbauelemente und/oder andere maschinen-lesbare Medien umfassen. Zusätzlich sind eine Reihe von Benutzer-Eingabe/Ausgabe-Einrichtungen, wie beispielsweise eine Tastatur 120 und eine Anzeige 125, mit dem Bus 115 gekoppelt. Der Prozessor 105 repräsentiert eine zentrale Verarbeitungseinheit einer beliebigen Architektur, wie beispielsweise CISC, RISC, VLIW oder Hybrid-Architektur. Außerdem kann der Prozessor 105 auf einem oder mehreren Chips implementiert sein. Der Bus 115 repräsentiert einen oder mehrere Busse (zum Beispiel AGP, PCI, ISA, X-Bus, VESA) und Brücken (auch als Bussteuereinrichtungen bezeichnet). Während dieses Ausführungsbeispiel unter Bezugnahme auf ein Einzelprozessor-Computersystem beschrieben wird, könnte die Erfindung auch in einem Multi-Prozessor-Computersystem implementiert werden.

Neben anderen Einrichtungen können wahlweise ein oder mehrere Netzwerke 130, ein TV-Signalempfänger 131, ein Fax/Modem 132, eine Digitalisiereinheit 133, eine Klingeinheit 134 und eine Graphikeinheit 135 mit dem Bus 115 gekoppelt sein. Die Netzwerkeinrichtung 130 und das Fax/Modem 132 repräsentieren eine oder mehrere Netzwerkverbindungen zum Übermitteln von Daten über ein maschinenlesbares Medium (z. B. Trägerwellen). Die Digitalisiereinheit 133 repräsentiert eine oder mehrere Einrichtungen zum Digitalisieren von Bildern (beispielsweise einen Scanner, eine Kamera, etc.). Die Klingeinheit 134 repräsentiert eine oder mehrere Einrichtungen zum Eingeben und/oder Ausgeben von Klängen (z. B. Mikrophone, Lautsprecher, Magnet-speicher, etc.). Die Graphikeinheit 135 repräsentiert eine oder mehrere Einrichtungen zum Erzeugen von 3D-Bildern (z. B. Graphikkarten). Fig. 1 veranschaulicht darüber hinaus, daß die Speichereinrichtung 110 Daten 136 und Software 137 speichert. Daten 136 repräsentieren Daten, die in einem oder mehreren der hier beschriebenen Formate gespeichert sind. Software 137 repräsentiert den notwendigen Befehlscode zum Durchführen irgendwelcher und/oder sämtlicher der unter Bezugnahme auf die Fig. 2 und 4 bis 6 beschriebenen Techniken. Selbstverständlich enthält die Speichereinrichtung 110 vorzugsweise (nicht gezeigte) zusätzliche Software, welche für das Verständnis der Erfindung nicht erforderlich ist.

Fig. 1 veranschaulicht darüber hinaus, daß der Prozessor 105 eine Dekodiereinheit 140, einen Registersatz 141, eine Ausführungseinheit 142 und einen internen Bus 143 zum Ausführen von Befehlen enthält. Der Prozessor 105 enthält ferner zwei interne Cache-Speicher, einen Ebene-0 (L0-) Cache-Speicher, welcher mit der Ausführungseinheit 142 gekoppelt ist, und einen Ebene-1 (L1-) Cache-Speicher, welcher mit dem L0-Cache gekoppelt ist. Ein externer Cache-Speicher, d. h. ein Ebene-2 (L2-) Cache-Speicher 172 ist mit dem Bus 115 über eine Cache-Steuereinrichtung 170 gekoppelt. Die tatsächliche Anordnung der verschiedenen Cache-Speicher ist eine Frage der Designauswahl oder kann durch die Computersystemarchitektur vorgegeben sein. So ist es klar, daß der L1-Cache auch außerhalb des Prozessors 105 angeordnet werden könnte. Bei alternativen Ausführungsbeispielen können mehr oder weniger Ebenen von Cache-

Speichern (anstelle L1 und L2) implementiert werden. In Fig. 1 sind drei Ebenen der Cache-Hierarchie gezeigt, aber es könnten mehr oder weniger Cache-Ebenen sein. Beispielsweise könnte die Erfindung auch dann ausgeführt werden, wenn es nur eine Cache-Ebene (nur L0) oder nur zwei Cache-Ebenen (L0 und L1) gäbe oder wenn es vier oder mehr Cache-Ebenen wären.

Selbstverständlich enthält der Prozessor 105 zusätzliche Schaltungen, welche für das Verständnis der Erfindung nicht erforderlich sind. Die Dekodiereinheit 140, Register 141 und Ausführungseinheit 142 sind miteinander über den internen Bus 143 gekoppelt. Die Dekodiereinheit 140 wird zum Dekodieren von durch den Prozessor 105 empfangenen Befehlen in Steuersignale und/oder Mikrobefehlscodeeintrittspunkte verwendet. In Abhängigkeit von diesen Steuersignalen und/oder Mikrobefehlscodeeintrittspunkten führt die Ausführungseinheit 142 die geeigneten Operationen durch. Die Dekodiereinheit 140 kann unter Verwendung einer beliebigen Anzahl unterschiedlicher Mechanismen implementiert werden (z. B. einer Nachschlagetabelle, einer Hardwareimplementierung, einer PLA, etc.). Während die Dekodierung der verschiedenen Befehle hier durch eine Serie von Wenn/dann-Aussagen repräsentiert wird, ist es klar, daß die Ausführung eines Befehls keine serielle Verarbeitung dieser Wenn/dann-Aussagen erfordert. Statt dessen wird jeder Mechanismus zum logischen Durchführen dieser Wenn/dann-Verarbeitung als innerhalb des Umfangs der Implementierung der Erfindung liegend angesehen.

Die gezeigte Dekodiereinheit 140 umfaßt eine Heranhol-einheit 150, welche Befehle heranholt, und einen Befehlsatz 155 zum Durchführen von Operationen an Daten. Der Befehlsatz 155 umfaßt erfindungsgemäße Cache-Steuere-befehle 166. Die Cache-Steuerebefehle 160 umfassen: einen Cache-Segment-Invalidier-Befehl, einen Cache-Segment-Flush-Befehl und einen Cache-Segment-Flush-und-Invali-dier-Befehl. Ein Beispiel des Cache-Segment-Invalidier-Befehls ist ein Seiten-Invalidier (PGINVD)-Befehl, der an einer vom Benutzer spezifizierten linearen Adresse operiert und die der linearen Adresse entsprechende physikalische 4KByte-Seite aus sämtlichen Ebenen der Cache-Hierarchie für sämtliche Teilnehmer in dem Computersystem, die mit dem Prozessor verbunden sind, invalidiert. Ein Beispiel des Cache-Segment-Flush-Befehls ist ein Seiten-Flush (PGFLUSH)-Befehl, der Daten in der der linearen Adresse entsprechenden physikalischen 4KByte-Seite einer Flush-Operation unterzieht. Ein Beispiel des Cache-Seg-ment-Flush-und-Invalidier-Befehls ist ein Seiten-Flush/In-validiert (PGFLUSHINV)-Befehl, der zunächst die Daten in der der linearen Adresse entsprechenden physikalischen 4KByte-Seite einer Flush-Operation unterzieht und dann die der linearen Adresse entsprechende physikalische 4KByte-Seite invalidiert. Bei alternativen Ausführungsbeispielen können die Cache-Steuerebefehle entweder an von einem Benutzer spezifizierten linearen oder physikalischen Adressen operieren und die zugehörigen Invalidier- und/oder Flush-Operationen gemäß den Prinzipien der Erfindung durchfüh-ren.

Zusätzlich zu den Cache-Segment-Invalidier-Befehlen, den Cache-Segment-Flush-Befehlen und den Cache-Seg-ment-Flush- und-Invalidier-Befehlen kann der Prozessor neue Befehle und/oder Befehle, die denen in vorhandenen Mehrzweckprozessoren zu findenden ähnlich sind, enthal-ten. Beispielsweise unterstützt der Prozessor 105 einen Be-fehlssatz, welcher mit dem Intel-Architektur-Befehlssatz kompatibel ist, der von vorhandenen Prozessoren verwendet wird, wie beispielsweise dem Pentium®-Prozessor.

Die Register 141 repräsentieren einen Speicherbereich des Prozessors 105 zum Speichern von Informationen, wie

beispielsweise Steuer/Status- Informationen, skalaren und/oder gepackten Ganzzahldaten, Gleitkommatdaten, etc. Es ist klar, daß ein Aspekt der Erfindung der beschriebene Befehlssatz ist. Gemäß diesem Aspekt der Erfindung ist der zum Speichern der Daten verwendete Speicherbereich unkritisch. Der Begriff Datenverarbeitungssystem wird hier für irgendeine Einrichtung zum Verarbeiten von Daten verwendet, einschließlich dem unter Bezugnahme auf Fig. 1 beschriebenen Prozessor umfaßt.

Fig. 2 veranschaulicht das Format des Cache-Segment-Invalidier-Befehls, des Cache-Segment-Flush-Befehls und des Cache-Segment-Flush-und-Invalidier-Befehls gemäß der Erfindung. Diese Befehle werden hier als Cache-Steuer-Befehle 160 bezeichnet. Die Cache-Steuerbefehle 160 umfassen einen Befehlscode (OP CODE) 210, welcher die Operation des Cache-Steuerbefehls 160 angibt, und einen Operanden 212, welcher den Namen eines Registers oder eines Speicherplatzes spezifiziert, welches bzw. welcher eine Startadresse des Datenobjekts hält, an welchem der Befehl 160 operieren wird.

Fig. 3 veranschaulicht die grundsätzliche Betriebsweise des Cache-Steuerbefehls 160. Bei der Ausführung der Erfindung stellt der Cache-Steuerbefehl 160 den Register-(oder Speicher-)Ort zur Verfügung, welcher eine Startadresse des Datenobjekts enthält, an dem der Befehl 160 operieren wird. Bei einem Ausführungsbeispiel umfaßt die Startadresse X am höchsten bewerteten Bits, welche in dem Register-(oder Speicher-)Ort gespeichert sind, und Y am geringsten bewerteten Bits. Der dem Cache-Steuerbefehl 160 zugeordnete Cache-Steuerprozeß verschiebt dann die X Bits nach rechts um Y Bit-Positionen, um die vollständige Startadresse zu erlangen. Dann arbeitet der Cache-Steuerbefehl 160 an den der Startadresse entsprechenden Daten in dem Cache-Speicher sowie an Daten, die Z nachfolgenden Adressen entsprechen. Bei einem Ausführungsbeispiel arbeitet der Cache-Steuerbefehl 160 an einer Seite von Daten, die im Cache-Speicher gespeichert ist, von welcher die Anfangsadresse in einem Register-(oder Speicher-)Ort gespeichert ist, der in dem Operanden 212 des Steuerbefehls spezifiziert ist. Bei alternativen Ausführungsbeispielen kann der Cache-Steuerbefehl 160 an einer beliebigen vorgegebenen, in dem Cache gespeicherten Datenmenge operieren, von welcher die Anfangsadresse in einem Register oder einem Speicherplatz gespeichert ist, das bzw. der von dem Operanden 212 in dem Cache-Steuerbefehl spezifiziert ist.

In Fig. 1 sind nur die L0-, L1- und L2-Ebenen gezeigt, aber es ist klar, daß mehr oder weniger Ebenen einfach implementiert werden können. Das in den Fig. 4 bis 6 gezeigte Ausführungsbeispiel beschreibt die Benutzung der Erfindung in bezug auf eine Cache-Ebene.

Details der verschiedenen Ausführungsbeispiele der Cache-Steuerbefehle 160 werden jetzt beschrieben. Zuerst wird der Cache-Segment-Invalidier-Befehl 162 beschrieben. Fig. 4A veranschaulicht ein Ausführungsbeispiel des Befehls. Bei Empfang des Cache-Segment-Invalidier-Befehls 162 bestimmt der Prozessor 105 aus dem Operanden 212 des Befehls 162 den Registerort, in welchem die am höchsten bewerteten Bits der Startadresse des Datenobjekts gespeichert sind. Dann verschiebt der Prozessor 105 den Wert in dem Operanden 212 um die Anzahl der am geringsten bewerteten Bits der Startadresse. Sobald die vollständige Startadresse gewonnen ist, setzt der Prozessor 105 die Invalidier-Bits des Cache-Speichers 300 der betroffenen Speicherplätze. Bei einem Ausführungsbeispiel wird eine Seite des Cache-Speichers 320 invalidiert, welche eine Startadresse aufweist, die der durch den Operanden 212 spezifizierten entspricht. Bei alternativen Ausführungsbeispielen werden unter Verwendung der vorliegenden Technik ir-

gendwelche vorgegebenen Abschnitte des Cache-Speichers 320 invalidiert, die eine Startadresse aufweisen, die derjenigen entspricht, die durch den Operanden 312 spezifiziert ist.

Fig. 4B zeigt ein Ausführungsbeispiel des Cache-Segment-Flush-Befehls 164. Bei Empfang des Cache-Segment-Flush-Befehls 164 bestimmt der Prozessor 105 aus dem Operanden 312 des Befehls 164 den Registerort, in welchem die am höchsten bewerteten Bits der Startadresse des Datenobjekts gespeichert sind. Dann verschiebt der Prozessor 105 den Wert durch die Anzahl der am geringsten bewerteten Bits der Startadresse. Sobald die vollständige Startadresse gewonnen ist, unterzieht der Prozessor diejenigen Speicherplätze des Cache-Speichers 320 der Flush-Operation, die durch die Ausführung des Befehls 164 betroffen sind. Bei einem Ausführungsbeispiel wird eine Seite des Cache-Speichers 320 der Flush-Operation unterzogen, die eine Startadresse aufweist, die derjenigen durch den Operanden 212 spezifizierten entspricht. Bei alternativen Ausführungsbeispielen werden Daten in irgendwelchen vorgegebenen Abschnitten des Cache-Speichers 320 der Flush-Operation unterzogen, die eine durch den Operanden 212 spezifizierte Startadresse aufweisen.

Fig. 4C veranschaulicht ein Ausführungsbeispiel des Cache-Segment-Flush-und-Invalidier-Befehls 166. Bei Empfang des Cache-Segment-Flush-und-Invalidier-Befehls 166 bestimmt der Prozessor 105 aus dem Operanden 212 des Befehls 166 den Registerort, an welchem die am höchsten bewerteten Bits der Startadresse des Datenobjekts gespeichert sind. Dann verschiebt der Prozessor 105 den Wert in dem durch den Operanden 212 spezifizierten Register um die Anzahl der am geringsten bewerteten Bits der Startadresse. Sobald die vollständige Startadresse gewonnen ist, unterzieht der Prozessor diejenigen Speicherorte des Cache-Speichers 320 einer Flush-Operation, die durch die Ausführung des Befehls 166 betroffen sind. Bei einem Ausführungsbeispiel wird eine Seite des Cache-Speichers 320 der Flush-Operation unterzogen. Bei einem alternativen Ausführungsbeispiel werden irgendwelche vorgegebenen Abschnitte des Cache-Speichers 320 der Flush-Operation unterzogen, die eine durch den Operanden 212 spezifizierte Startadresse aufweisen. Als nächstes invalidiert der Prozessor 105 die betroffenen Bereiche des Cache-Speichers 320, die der Flush-Operation unterzogen wurden. Bei einem Ausführungsbeispiel wird dies durch Setzen der Invalidier-Bits jeder betroffenen Cache-Zeile durchgeführt.

Fig. 5A ist ein Ablaufdiagramm, das ein Ausführungsbeispiel des Cache-Segment-Invalidier-Prozesses gemäß der Erfindung veranschaulicht. Beginnend beim Start-Zustand fährt der Prozeß 500 zum Verarbeitungsblock 510 fort, wo er den Operanden 212 des von dem Prozessor 105 empfangenen Befehls 162 überprüft, um den Speicherort des Werts zu bestimmen, der die am höchsten bewerteten Bits der Startadresse der zugehörigen Operation darstellt. Der Prozeß 500 fährt dann zum Verarbeitungsblock 512 fort, wo er den die am höchsten bewerteten Bits der Startadresse repräsentierenden Wert aus dem spezifizierten Speicherort gewinnt. Der Prozeß 500 schreitet dann zum Verarbeitungsblock 514 fort, wo er den gewonnenen Wert um eine vorgegebene Anzahl von Bits verschiebt. Bei einem Ausführungsbeispiel repräsentiert die vorgegebene Anzahl die Anzahl der am geringsten bewerteten Bits in der Startadresse. Als nächstes bestimmt der Prozeß 500 das Cache-Segment, das durch die Operation bzw. den Befehl 162 betroffen ist, wie es im Verarbeitungsblock 516 gezeigt ist. Bei einem Ausführungsbeispiel ist das Cache-Segment eine Seite. Bei einem Ausführungsbeispiel enthält eine Seite 4KBytes. Bei alternativen Ausführungsbeispielen kann das Cache-Segment ein beliebiger vorgegebener Abschnitt des Cache-

Speichers sein. Der Prozeß 500 fährt dann zum Verarbeitungsblock 518 fort, wo er die Daten in dem zugehörigen Cache-Segment beginnend an der spezifizierten Startadresse invalidiert. Bei einem Ausführungsbeispiel wird dies ausgeführt, indem die jeder Cache-Zeile in dem Cache-Segment entsprechenden Ungültig-Bits oder Invalidier-Bits gesetzt werden. Dann endet der Prozeß 500.

Fig. 5B ist ein Ablaufdiagramm, das ein Ausführungsbeispiel des erfindungsgemäßen Cache-Segment-Flush-Prozesses veranschaulicht. Beginnend im Start-Zustand fährt der Prozeß 520 zum Verarbeitungsblock 522 fort, wo er den Operanden 212 des von dem Prozessor 105 empfangenen Befehls 164 oder 166 überprüft, um den Speicherort desjenigen Werts zu bestimmen, der die am höchsten bewerteten Bits der Startadresse der zugehörigen Operation repräsentiert. Der Prozeß 520 fährt dann zum Verarbeitungsblock 524 fort, wo er den die am höchsten bewerteten Bits der Startadresse repräsentierenden Wert aus dem spezifizierten Speicherort gewinnt. Der Prozeß 520 schreitet dann zum Verarbeitungsblock 526 weiter, wo er den gewonnenen Wert um eine vorgegebene Anzahl von Bits verschiebt. Bei einem Ausführungsbeispiel repräsentiert die vorgegebene Anzahl die Anzahl der am geringsten bewerteten Bits in der Startadresse. Als nächstes bestimmt der Prozeß 520 das von der Operation bzw. den Befehlen 164 oder 166 betroffene Cache-Segment, wie es im Verarbeitungsblock 528 gezeigt ist. Bei einem Ausführungsbeispiel ist das Cache-Segment eine Seite. Bei alternativen Ausführungsbeispielen kann das Cache-Segment ein beliebiger vorgegebener Abschnitt des Cache-Speichers sein. Der Prozeß 520 fährt dann zum Verarbeitungsblock 530 fort, wo er den Inhalt des spezifizierten Cache-Segments in die Speichereinrichtung 110 spült (flush). Der Prozeß 520 fährt dann zum Entscheidungsblock 532 fort, wo er abfragt, ob der empfangene Befehl ein Flush-Befehl oder ein Flush-und-Invalidier-Befehl ist. Sofern der Befehl ein Flush-Befehl ist, endet der Prozeß 520. Sofern der Befehl ein Flush-und-Invalidier-Befehl ist, fährt der Prozeß 520 mit dem Verarbeitungsblock 534 fort, wo er die Daten in dem zugehörigen Cache-Segment beginnend an der spezifizierten Startadresse invalidiert. Bei einem Ausführungsbeispiel wird dies durchgeführt, indem die jeder Cache-Zeile im Cache-Segment entsprechenden Ungültig-Bits bzw. Invalidier-Bits gesetzt werden. Dann endet der Prozeß 520.

Die Verwendung der vorliegenden Erfindung verbessert somit die Systemleistung, indem ein Invalidier-Befehl und/oder ein Flush-Befehl zum Invalidieren und/oder Spülen von Daten in einem beliebigen vorgegebenen Abschnitt des Cache-Speichers zur Verfügung gestellt wird. In Fällen, wo die Konsistenz zwischen dem Cache-Speicher und dem Hauptspeicher durch Software aufrechterhalten wird, wird die Systemleistung verbessert, da eine Flush-Operation nur der betroffenen Abschnitte des Cache-Speichers effektiver und flexibler ist als das Spülen des gesamten Cache-Speichers. Außerdem wird die Systemleistung verbessert, indem Flush- und/oder Invalidier-Operationen zur Verfügung gestellt werden, die eine größere Granularität als eine Cache-Zeilengröße haben, da der Benutzer unter Verwendung eines einzigen Befehls einen Speicherbereich einer Flush- und/oder Invalidier-Operation unterziehen kann, ohne daß er den Befehlscode ändern muß, wenn das Computersystem die Größe einer Cache-Zeile ändert.

#### Patentansprüche

1. Computersystem, aufweisend:  
einen Cache-Speicher mit einer Mehrzahl von Daten speichernden Cache-Zeilen;

einen Speicherbereich zum Speichern eines Datenoperanden; und

einer mit dem Speicherbereich gekoppelten Ausführungseinheit, die in Abhängigkeit vom Empfang eines einzelnen Befehls an Datenelementen in dem Datenoperanden operiert und Daten in einem vorgegebenen Abschnitt der Mehrzahl von Cache-Zeilen invalidiert.

2. Computersystem nach Anspruch 1, dadurch gekennzeichnet, daß der Datenoperand ein Registerort ist.

3. Computersystem nach Anspruch 2, dadurch gekennzeichnet, daß der Registerort einen Abschnitt einer Startadresse der Cache-Zeile enthält, in welcher Daten invalidiert werden sollen.

4. Computersystem nach Anspruch 3, dadurch gekennzeichnet, daß der Abschnitt der Startadresse mehrere am höchsten bewertete Bits der Startadresse enthält.

5. Computersystem nach Anspruch 4, dadurch gekennzeichnet, daß die Ausführungseinheit die Datenelemente um eine vorgegebene Anzahl von Bitpositionen verschiebt, um die Startadresse der Cache-Zeile zu gewinnen, in welcher Daten invalidiert werden sollen.

6. Computersystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß der vorgegebene Abschnitt der Mehrzahl von Cache-Zeilen eine Seite in dem Cache-Speicher ist.

7. Computersystem, aufweisend:

einen ersten Speicherbereich zum Speichern von Daten;

einen Cache-Speicher mit einer Mehrzahl von Daten speichernden Cache-Zeilen;

einen zweiten Speicherbereich zum Speichern eines Datenoperanden, und

eine mit dem ersten Speicherbereich, dem zweiten Speicherbereich und dem Cache-Speicher gekoppelte Ausführungseinheit, die in Abhängigkeit vom Empfang eines einzelnen Befehls an Datenelementen in dem Datenoperanden operiert, um Daten von einem vorgegebenen Abschnitt der Mehrzahl von Cache-Zeilen in dem Cache-Speicher in den ersten Speicherbereich zu kopieren.

8. Computersystem nach Anspruch 7, dadurch gekennzeichnet, daß der Datenoperand ein Registerort ist.

9. Computersystem nach Anspruch 8, dadurch gekennzeichnet, daß der Registerort eine Mehrzahl von am höchsten bewerteten Bits einer Startadresse derjenigen Cache-Zeile enthält, aus welcher die Daten kopiert werden sollen.

10. Computersystem nach Anspruch 9, dadurch gekennzeichnet, daß die Ausführungseinheit die Datenelemente um eine vorgegebene Anzahl von Bitpositionen verschiebt, um die Startadresse der Cache-Zeile zu gewinnen, aus welcher die Daten kopiert werden sollen.

11. Computersystem nach einem der Ansprüche 7 bis 10, dadurch gekennzeichnet, daß der vorgegebene Abschnitt der Mehrzahl von Cache-Zeilen eine Seite in dem Cache-Speicher ist.

12. Computersystem nach einem der Ansprüche 7 bis 11, dadurch gekennzeichnet, daß die Ausführungseinheit darüber hinaus in Erwiderung des Empfangs eines einzigen Befehls Daten in dem vorgegebenen Abschnitt der Mehrzahl von Cache-Zeilen nach dem Kopieren der Daten in den ersten Speicherbereich invalidiert.

13. Prozessor, aufweisend:

einen Dekodierer zum Dekodieren von Befehlen, und eine mit dem Dekodierer gekoppelte Schaltung, die in

Erwiderung eines einzelnen dekodierten Befehls:  
 eine Startadresse eines vorgegebenen Bereichs eines  
 Cache-Speichers gewinnt, an welchem der Befehl  
 durchgeführt werden wird;  
 Daten in dem vorgegebenen Bereich des Cache-Spei- 5  
 chers invalidiert.

14. Prozessor nach Anspruch 13, dadurch gekenn-  
 zeichnet, daß ein Abschnitt der Startadresse in einem in  
 dem dekodierten Befehl spezifizierten Register ange-  
 ordnet ist. 10

15. Prozessor nach Anspruch 14, dadurch gekenn-  
 zeichnet, daß der Abschnitt der Startadresse eine Mehr-  
 zahl der am höchsten bewerteten Bits der Startadresse  
 umfaßt.

16. Prozessor nach Anspruch 15, dadurch gekenn- 15  
 zeichnet, daß die Schaltung die Datenelemente um eine  
 vorgegebene Anzahl von Bit-Positionen verschiebt, um  
 die Startadresse des Cache-Bereichs zu gewinnen, in  
 welchem die Daten invalidiert werden sollen.

17. Prozessor nach einem der Ansprüche 13 bis 16, da- 20  
 durch gekennzeichnet, daß der vorgegebene Bereich  
 des Cache-Speichers eine Seite in dem Cache-Speicher  
 ist.

18. Prozessor, aufweisend:  
 einen Dekodierer zum Dekodieren von Befehlen, und 25  
 eine mit dem Dekodierer gekoppelte Schaltung, die in  
 Erwiderung eines einzelnen dekodierten Befehls:  
 eine Startadresse eines vorgegebenen Bereichs eines  
 Cache-Speichers gewinnt, an welchem der Befehl  
 durchgeführt wird; 30  
 Daten aus dem vorgegebenen Bereich des Cache-Spei-  
 chers kopiert; und  
 die kopierten Daten in einem von dem Cache-Speicher  
 getrennten Speicherbereich speichert.

19. Prozessor nach Anspruch 18, dadurch gekenn- 35  
 zeichnet, daß ein Abschnitt der Startadresse in einem in  
 dem dekodierten Befehl spezifizierten Register ange-  
 ordnet ist.

20. Prozessor nach Anspruch 19, dadurch gekenn- 40  
 zeichnet, daß der Abschnitt der Startadresse eine Mehr-  
 zahl von am höchsten bewerteten Bits der Startadresse  
 umfaßt.

21. Prozessor nach Anspruch 20, dadurch gekenn- 45  
 zeichnet, daß die Schaltung die Datenelemente um eine  
 vorgegebene Anzahl von Bitpositionen verschiebt, um  
 die Startadresse des Cache-Bereichs zu gewinnen, aus  
 welchem die Daten kopiert werden sollen.

22. Prozessor nach einem der Ansprüche 18 bis 21, da-  
 durch gekennzeichnet, daß der vorgegebene Bereich  
 des Cache-Speichers eine Seite in dem Cache-Speicher 50  
 ist.

23. Prozessor nach einem der Ansprüche 18 bis 22, da-  
 durch gekennzeichnet, daß die Schaltung ferner die  
 Daten in dem vorgegebenen Bereich des Cache-Spei- 55  
 chers in Erwiderung des Empfangs des einzelnen Be-  
 fehls nach Kopieren der Daten in dem Speicherbereich  
 invalidiert.

24. Computer-implementiertes Verfahren, wobei:  
 a) ein einzelner Befehl dekodiert wird;  
 b) in Erwiderung des Dekodierens des einzelnen 60  
 Befehls eine Startadresse eines vorgegebenen Be-  
 reichs eines Cache-Speichers gewonnen wird, an  
 welchem Bereich der einzelne Befehl durchge-  
 führt wird; und  
 c) die Ausführung des einzelnen Befehls abge- 65  
 schlossen wird, indem Daten in dem vorgegeben-  
 en Bereich des Cache-Speichers invalidiert wer-  
 den.

25. Verfahren nach Anspruch 24, dadurch gekenn-  
 zeichnet, daß beim Invalidieren Ungültig-Bits in dem  
 vorgegebenen Bereich des Cache-Speichers gesetzt  
 werden.

26. Verfahren nach Anspruch 24 oder 25, dadurch gekenn-  
 zeichnet, daß zum Gewinnen der Startadresse:  
 b.1) ein Abschnitt der Startadresse aus einem in  
 dem dekodierten Befehl spezifizierten Speicher-  
 platz gewonnen wird;  
 b.2) der Abschnitt der Startadresse um eine vorge-  
 gebene Anzahl von Positionen verschoben wird.

27. Verfahren nach Anspruch 26, dadurch gekenn-  
 zeichnet, daß im Schritt b.1) der Abschnitt der Start-  
 adresse eine Mehrzahl von am höchsten bewerteten  
 Bits der Startadresse enthält, und daß im Schritt b.2)  
 die vorgegebene Anzahl von Bitpositionen die Anzahl  
 der am geringsten bewerteten Bits der Startadresse dar-  
 stellt.

28. Verfahren nach einem der Ansprüche 24 bis 27,  
 dadurch gekennzeichnet, daß der vorgegebene Bereich  
 des Cache-Speichers eine Seite in dem Cache-Speicher  
 ist.

29. Computer-implementiertes Verfahren, wobei:  
 a) ein einzelner Befehl dekodiert wird;  
 b) in Erwiderung des Dekodierens des einzelnen  
 Befehls eine Startadresse eines vorgegebenen Be-  
 reichs eines Cache-Speichers gewonnen wird, an  
 welchem Bereich der einzelne Befehl durchge-  
 führt wird; und  
 c) die Ausführung des einzelnen Befehls abge-  
 schlossen wird, indem Daten aus dem vorgegeben-  
 en Bereich des Cache-Speichers kopiert und die  
 kopierten Daten in einem von dem Cache-Spei-  
 cher getrennten Speicherbereich gespeichert wer-  
 den.

30. Verfahren nach Anspruch 29, dadurch gekenn-  
 zeichnet, daß im Schritt b):  
 b.1) ein Abschnitt der Startadresse aus einem in  
 dem dekodierten Befehl spezifizierten Speicher-  
 platz gewonnen wird;  
 b.2) der Abschnitt der Startadresse um eine vorge-  
 gebene Anzahl von Bitpositionen verschoben  
 wird, um die Startadresse des Cache-Bereichs zu  
 erlangen, aus welchem Daten kopiert werden sol-  
 len.

31. Verfahren nach Anspruch 30, dadurch gekenn-  
 zeichnet, daß im Schritt b.1) der Abschnitt der Start-  
 adresse eine Mehrzahl von am höchsten bewerteten  
 Bits der Startadresse enthält, und daß im Schritt b.2)  
 die vorgegebene Anzahl von Bitpositionen die Anzahl  
 der am geringsten bewerteten Bits der Startadresse re-  
 präsentiert.

32. Verfahren nach einem der Ansprüche 29 bis 31,  
 dadurch gekennzeichnet, daß der vorgegebene Bereich  
 des Cache-Speichers eine Seite in dem Cache-Speicher  
 ist.

33. Verfahren nach einem der Ansprüche 29 bis 32,  
 dadurch gekennzeichnet, daß  
 d) die Daten in dem vorgegebenen Bereich des  
 Cache-Speichers in Erwiderung des Empfangs  
 des einzelnen Befehls nach dem Kopieren der Da-  
 ten zu dem Speicherbereich invalidiert werden.

34. Computer-lesbare Einrichtung, aufweisend:  
 ein computer-lesbares Medium, das einen Befehl spei-  
 chert, welcher dann, wenn er von einem Prozessor aus-  
 geführt wird, den Prozessor veranlaßt:  
 eine Startadresse eines vorgegebenen Bereichs eines  
 Cache-Speichers, in welchem der Befehl durchgeführt

wird, zu gewinnen; und

Daten in dem vorgegebenen Bereich des Cache-Speichers zu invalidieren.

35. Computer-lesbare Einrichtung, aufweisend:

ein computer-lesbares Medium, das einen Befehl speichert, welcher dann, wenn er von einem Prozessor ausgeführt wird, den Prozessor veranlaßt:

eine Startadresse eines vorgegebenen Bereichs eines Cache-Speichers, in welchem der Befehl durchgeführt wird, zu gewinnen;

Daten aus dem vorgegebenen Bereich des Cache-Speichers zu kopieren; und

die kopierten Daten in einem von dem Cache-Speicher getrennten Speicherbereich zu speichern.

36. Einrichtung nach Anspruch 35, dadurch gekennzeichnet, daß der Befehl ferner den Prozessor veranlaßt, die Daten in dem vorgegebenen Bereich des Cache-Speichers nach dem Kopieren der Daten in den Speicherbereich zu invalidieren.

---

Hierzu 8 Seite(n) Zeichnungen

---

20

25

30

35

40

45

50

55

60

65

- Leerseite -

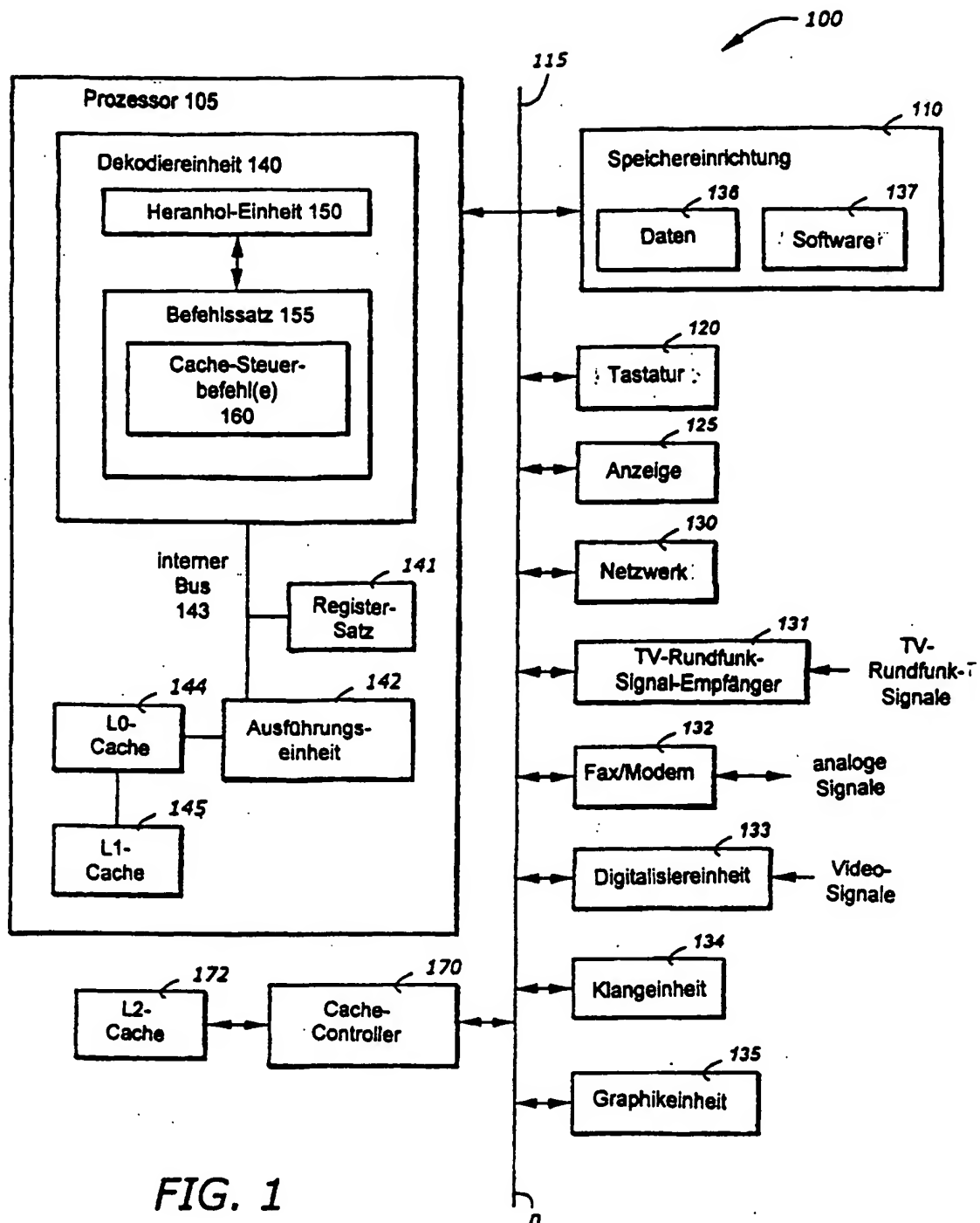


FIG. 1

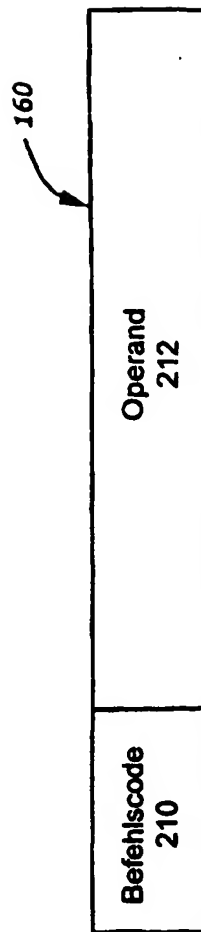


FIG. 2

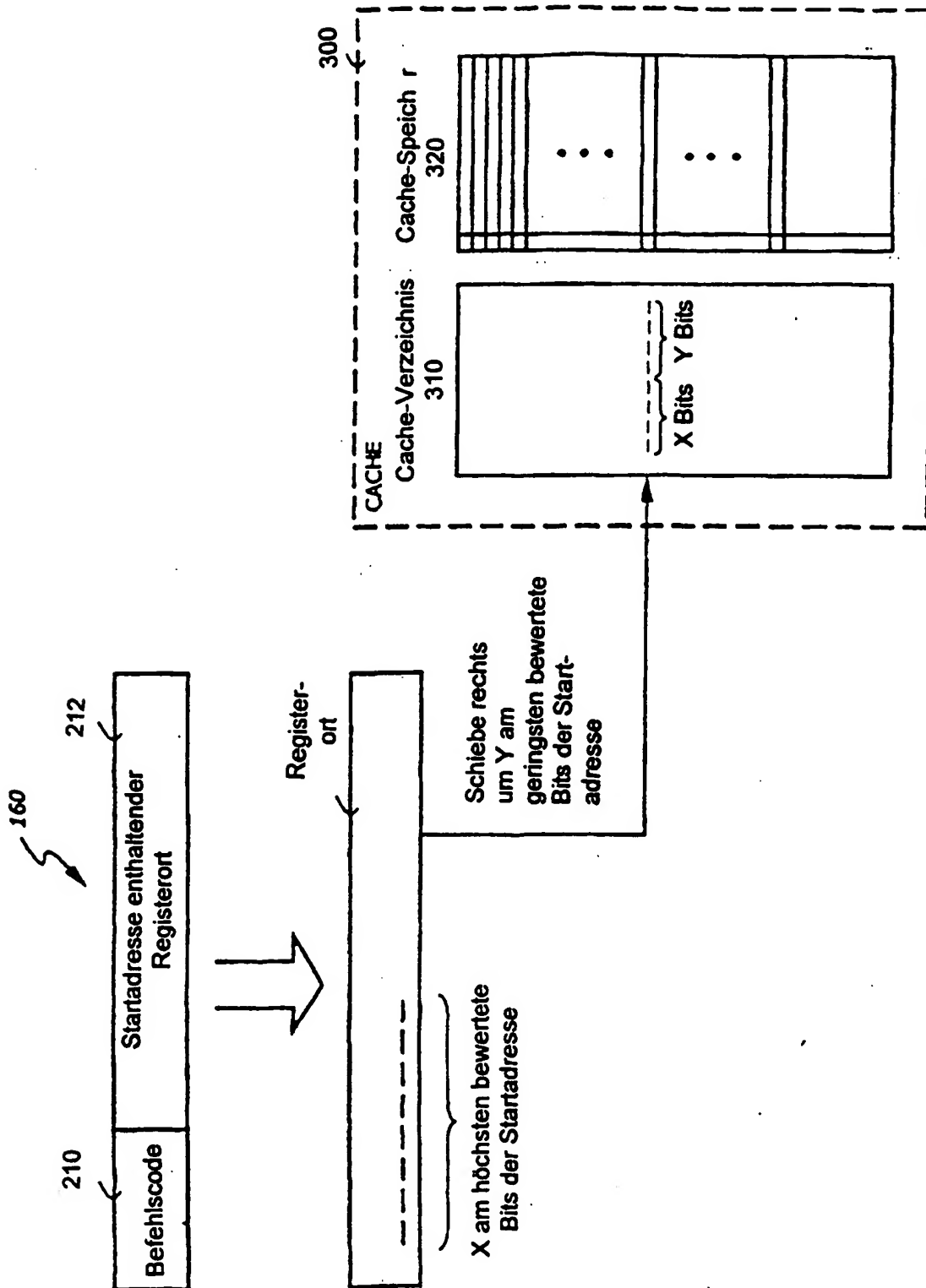
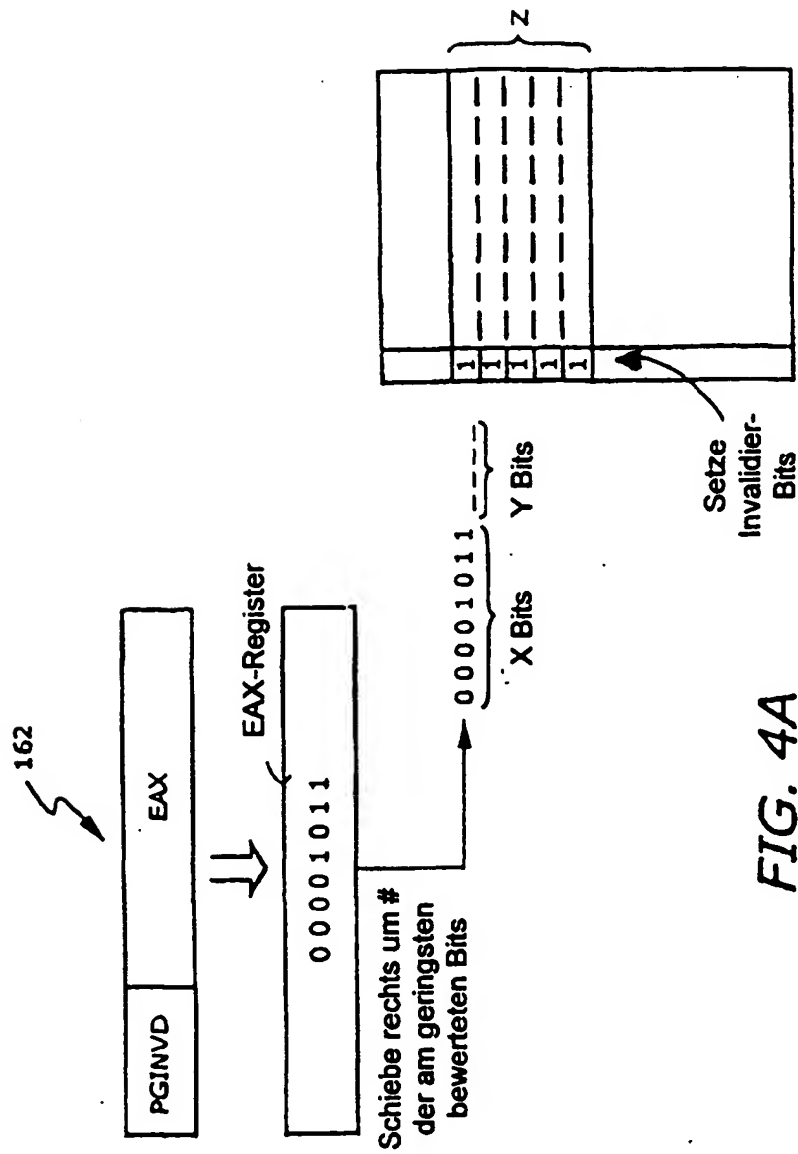


FIG. 3



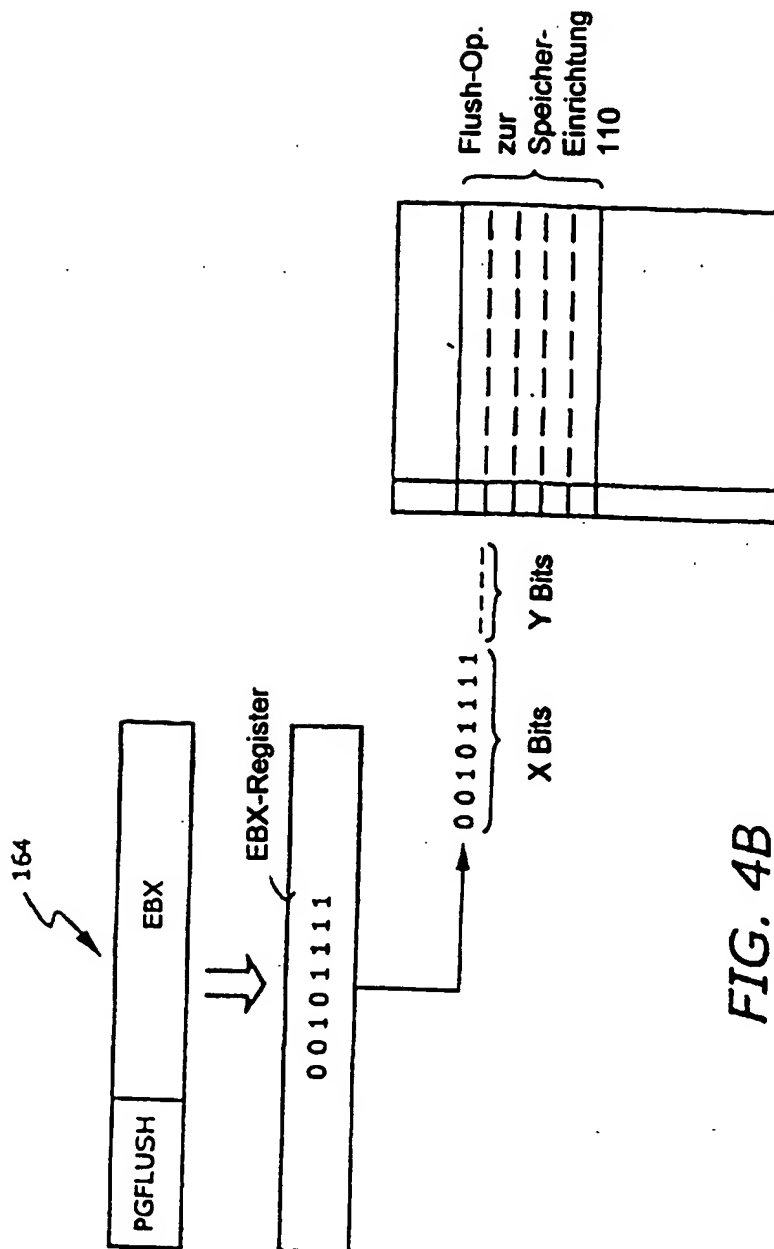


FIG. 4B

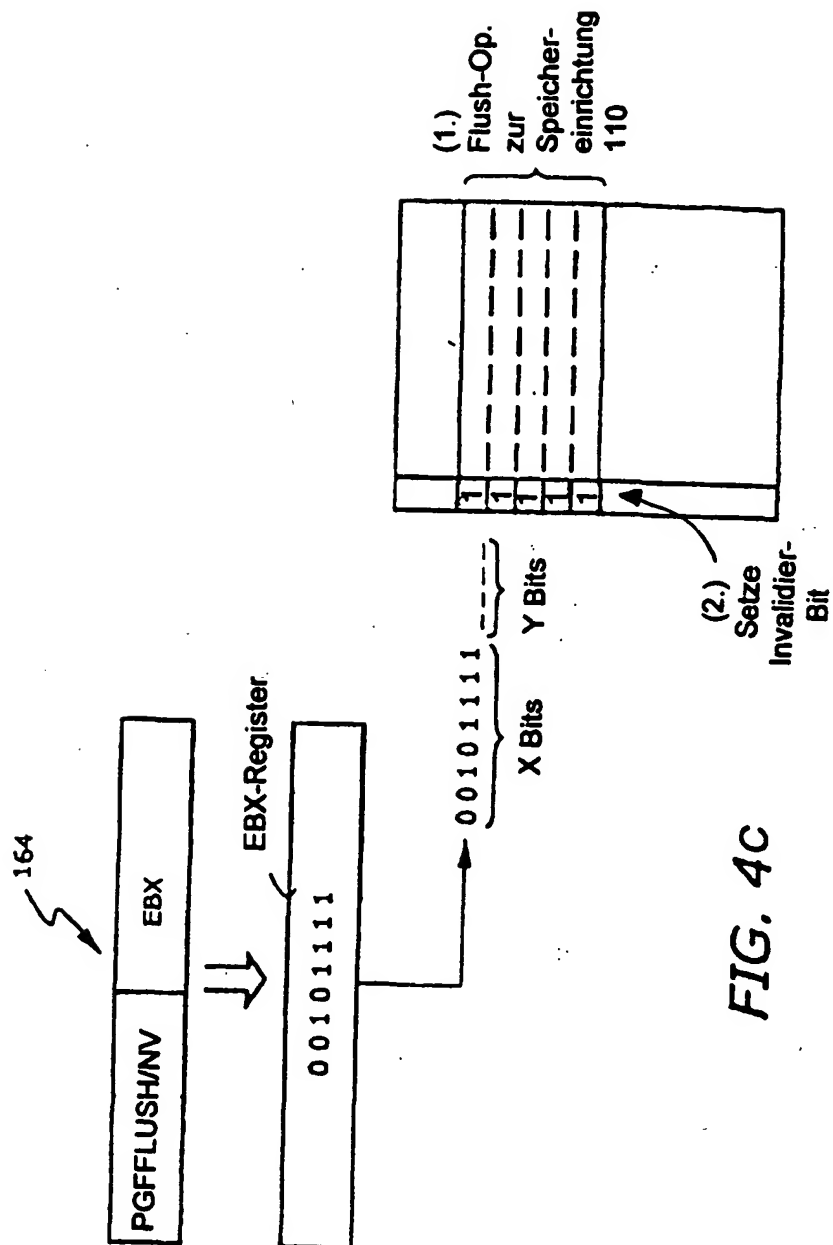


FIG. 4C

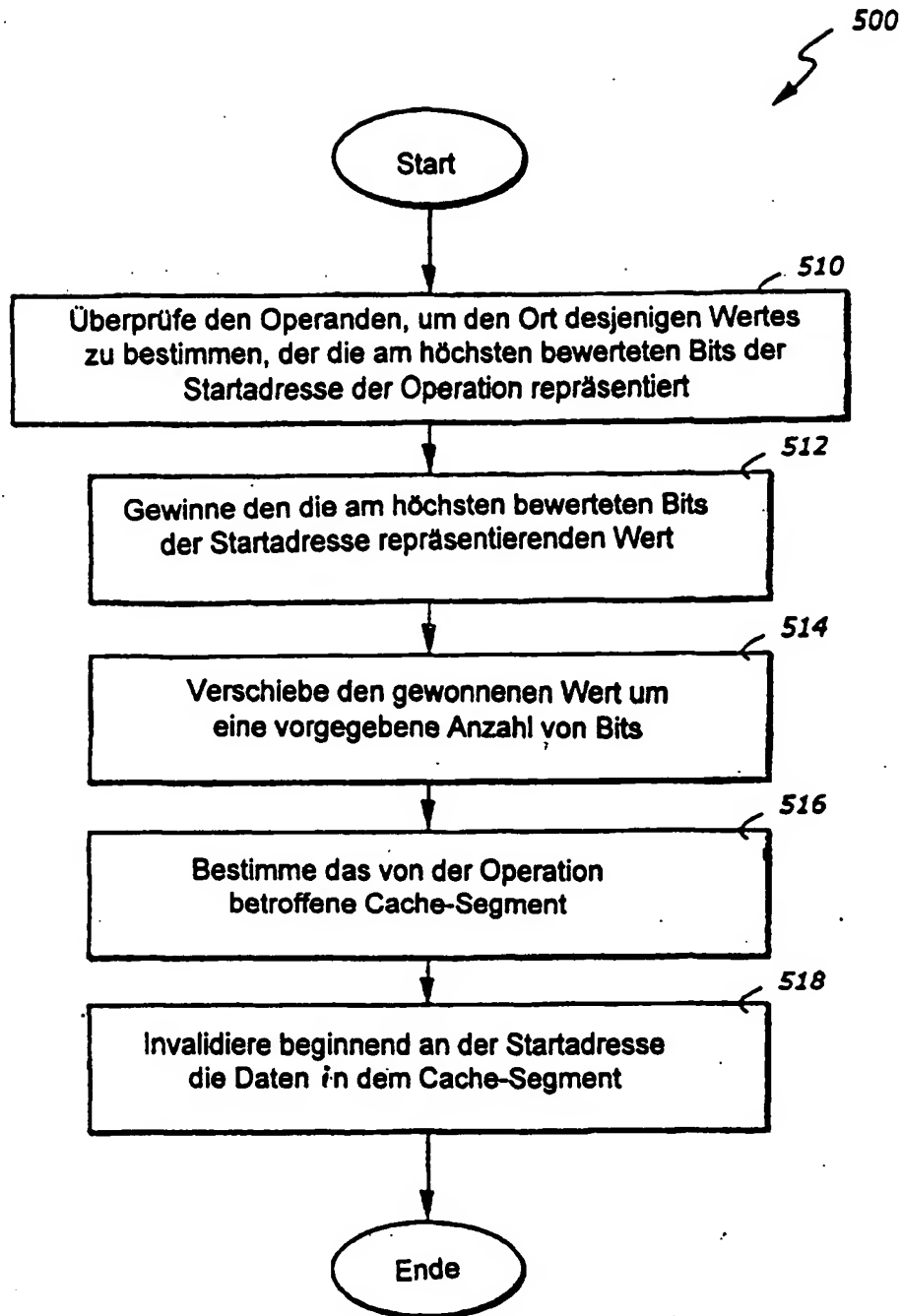


FIG. 5A

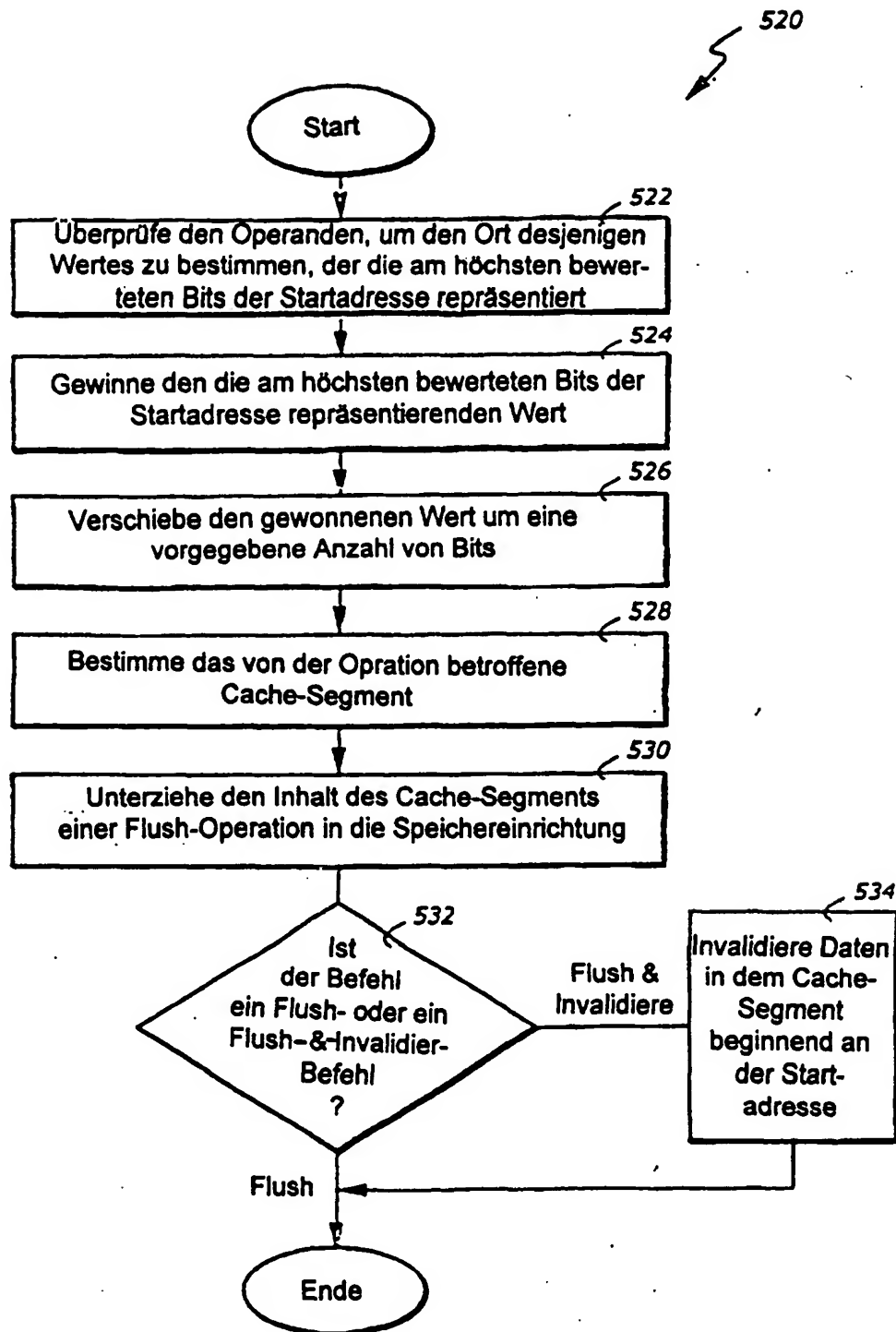


FIG. 5B

Abstract (Basic): \*DB 19934515\* A1

NOVELTY - The computer system includes a cache-store with several data storage cache lines. There is a storage area for saving data operands, and a processing unit coupled to the store. This operates according to the received single commands and data elements in the operands, and invalidates data in a predetermined region of the cache lines.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for a computer implemented method and a computer readable unit.

USE - For conducting cache-segment flush and invalidation operations.

ADVANTAGE - Overcomes certain disadvantages of conventional methods and systems, e.g. that a data error can occur when a cache invalidation is only possible on the entire cache.

DESCRIPTION OF DRAWING(S) - The figure shows a flow diagram of an example cache segment invalidation process.

pp; 15 DwgNo 5A/5

Title Terms: COMPUTER; SYSTEM; CONDUCTING; CACHE; SEGMENT; FLUSH; INVALID;  
OPERATE